CLAIMS

We claim:

1. A computer readable medium containing a data structure for saving state for a semantically accessible state binding method, the computer readable medium comprising:

a first state frame including a representation of a state of an executing program; and

a second state frame including a representation of state changes made by the executing program after the first state frame is created and the second state frame includes a pointer back to the first state frame.

2. The computer readable medium of claim 1 further comprising:

a third state frame including a representation of state changes made by the executing program after a fork method creates the third frame, and the third state frame includes a pointer back to the second frame.

3. The computer readable medium of claim 2 wherein a fourth frame includes changes made by the executing program after the fork method creates the third state frame and after a set method returns the executing program to the state of the second state frame.

4. The computer readable medium of claim 3 further comprising a joined state frame including a combination of state changes in the third and fourth frames.

5. The computer readable medium of claim 3 wherein a first thread of the executing program makes state changes copied in the second frame, and a second thread of the executing program makes state changes copied into the third frame.

6. The computer readable medium of claim 1 wherein the second state frame includes unchanged state read from the first state frame.

7. A computerized method comprising:

receiving via an application programming interface a request to create a state save;

saving a first representation of a state of an executing program in response to the request;

maintaining a second representation of subsequent state comprising changes made to the state of the executing program after the first representation; and

resetting the executing program to the saved first representation upon receiving a state set request at the application programming interface.

8. A computer system comprising:

memory and a central processing unit executing,

a program including executable instructions and an evolving present state; and

a state component comprising:

an initial representation of a prior evolving present state of the program,

a subsequent representation of state changes made by the program since the initial representation, and

a method for returning the program state to the prior evolving present state.

9. The computer system of claim 8 wherein the state component further includes a method for reading a location and value from the initial representation of the

prior evolving present state and storing the value in a subsequent representation of the state changes.

10. The computer system of claim 8, wherein the state component includes a fork method for maintaining state for a thread spawned by the program and a forked representation of state changes made by the spawned thread of the program.

11. The system of claim 10 wherein the state component includes a join method for joining state changes made by the forked thread back into state changes of the subsequent representation.

12. A method comprising:

executing a program which executes an executable model comprising a transforming present model state and actions causing changes to the present model state;

saving a present state of the executable model in response to a state save request received from the program via an application programming interface;

setting the present model state of the executable model to the saved present state in response to a state set request received from the program via the application programming interface; and

after setting the present model state, executing another action causing changes to the set present model state.

13. The method of claim 12 wherein the executing model exercises plural actions, each action exercised after setting the present model state to the saved present state.

14. The method of claim 13 wherein the plural actions are recorded thereby creating action sequences used to exercise a state space of a program that the executable model is designed to model.

15. A computer system comprising:

memory and a central processing unit executing,

a program comprising instructions and a state changing relative to time, and

a state mechanism linked into the program and providing

a save state reference the program accesses to save a present changing state,

a service that creates a current state record and saves a representation of state changes made by the program after the program saves a present changing state via the reference, and

a state set reference the program accesses to reset the program state to a prior saved present changing state.

16. The system of claim 15, wherein the state mechanism further provides a service that reads a value stored in a prior saved state and copies the read value into the current state record.

17. A computer readable medium comprising computer executable instructions for performing the method of claim 7.

18. A computer readable medium comprising computer executable instructions for performing the method of claim 12.

19. A computer readable method comprising computer executable instructions for performing a method comprising:

receiving a request to create a saved state of an executing model;

saving a first representation of a state of the executing model;

maintaining a second representation of state changes made by the executing model after the first representation; and

reinstating the executing model state to the state of the first representation.

20. A computerized method comprising:

executing a program comprising a transforming present state;

saving a present state of the program in response to a state save request received from the program via an application programming interface; and

after the program transforms state from the saved present state, setting the present state of the program to the saved present state in response to a state set request received from the program via the application programming interface.

21. The method of claim 20 wherein the program varies an execution path after setting the present state to the saved present state.